CLAIMS

We claim:

1. A method for decreasing a computer application start-up time, comprising:

creating a serialized representation of application objects in a runtime environment;

building an object code file using the serialized representation; and

providing the application to a new runtime environment.

2. The method of claim 1 wherein the step of creating includes:

reading from a runtime memory space a description of each object of a running application.

3. The method of claim 1 wherein the step of creating includes enumerating a description of each object of the application using reflection

4. The method of claim 2 wherein the runtime environment is a virtual machine.

5. The method of claim 4 wherein the virtual machine is a Flash renderer.

6. The method of claim 1 wherein the step of creating comprises:

identifying each object of a running application by a unique identifier.

7. The method of claim 6 wherein the step of creating comprises detaching each object from an object hierarchy and creating a description of each slot in said object.

8. The method of claim 6 wherein the step of creating includes the step of providing the representation directly to the building step.

9. The method of claim 1 wherein the serialized application is written in an extensible markup language data format.

10. The method of claim 9 wherein the step of creating comprises storing the markup language file prior to said step of building.

11. The method of claim 1 wherein the step of creating comprises assigning a serialization identifier to each object.

12. The method of claim 1 further including the step of developing the application in an interpreted language.

13. The method of claim 6 wherein the step of serializing comprises:

assigning a function ID to each function in the application.

14. The method of claim 13 wherein the method further includes:

creating a function ID table associating each function ID with function code.

15. The method of claim 13 wherein the method further includes assigning unique function identifiers to functions within closures.

16. The method of claim 1 wherein the step of building comprises using a compiled version of application source compiled prior to the creating step in combination with the serialized representation to build the new object code file.

17. The method of claim 1 wherein the step of creating comprises:
writing the serialized description to a text file prior to said step of building.

18. The method of claim 1 wherein the step of creating is performed in the runtime.

19. The method of claim 1 wherein the step of creating is performed in a different runtime.

20. A method for providing an optimized application, comprising:
compiling an application provided in an source language;
initializing the application in a runtime environment; and
creating a serialized representation of the application.

21. The method of claim 20 wherein the step of creating comprises:
reading from the runtime environment a description of each object of the running application.

22. The method of claim 21 wherein the step of creating further comprises:
outputting the description to a rebuilder.

23. The method of claim 22 wherein the step of outputting comprises storing the serialized representation in a text file and providing the file to the rebuilder.

24. The method of claim 22 wherein the step of outputting comprises:

writing the description to an Extensible Markup Language file and providing the file to the rebuilder.

25. The method of claim 20 wherein the runtime environment is a virtual machine.

26. The method of claim 20 wherein the virtual machine is a Flash renderer.

27. The method of claim 20 wherein the step of creating includes assigning a serialization identifier to each initialized object.

28. The method of claim 20 wherein the step of creating includes the steps of enumerating each object in a global scope and writing a serialized description of each said object.

29. The method of claim 20 wherein the step of serializing comprises:

assigning a function ID to each function in the application.

30. The method of claim 29 wherein the method further includes:

creating a function ID table associating each function ID with a function call.

31. The method of claim 29 wherein the method further includes assigning function identifiers to functions within closures.

32. The method of claim 20 further including the step of combining the serialized file with an object code file crated by said compiling step to create a new object code file.

33. A method of operating an application, comprising:

requesting an application from an application source server; and

receiving object code of a serialized description of the application from the source server, the object code including instructions for creating a runtime memory state of the application.

34. The method of claim 33 wherein the object code includes media assets.

35. The method of claim 33 wherein the runtime memory state is of a Flash renderer.

36. The method of claim 33 wherein the step of receiving includes the step of:

loading the object code into the runtime; and

executing the object code.

37. The method of claim 33 wherein the step of serializing comprises:

calling a function using a function identifier from the object code; and

receiving function code from the source server.

38. One or more processor readable storage devices having processor readable code embodied on said processor readable storage devices, said processor readable code for programming one or more processors to perform a method comprising the steps of:

creating a serialized representation of application objects in a runtime environment;

building an object code file using the serialized representation; and

providing the application to a new runtime environment.

39. One or more processor readable storage devices as described in claim 38 wherein the step of creating includes:

reading from a runtime environment memory space a description of each object of a running application.

40. One or more processor readable storage devices as described in claim 38 wherein the step of creating includes enumerating a description of each object of the application using reflection.

41. One or more processor readable storage devices as described in claim 38 wherein the step of creating comprises:

identifying each object of a running application by a unique identifier.

42. One or more processor readable storage devices as described in claim 41 wherein the step of creating comprises:

writing the description to a text file and compiling the text file.

43. One or more processor readable storage devices as described in claim 41 wherein the step of creating comprises:

writing the description to an Extensible markup language file and compiling the markup language file.

44. One or more processor readable storage devices as described in claim 41 wherein the step of creating comprises detaching each object from an object hierarchy and creating a description of each slot in said object.

45. One or more processor readable storage devices as described in claim 41 wherein the serialization step further includes the step of:

determining whether the object is a class; and

writing a serialized description of the class.

46. One or more processor readable storage devices as described in claim 39 wherein the serialized application is written in an extensible markup language data format.

47. One or more processor readable storage devices as described in claim 39 wherein the step of creating comprises assigning a serialization identifier to each object.

48. One or more processor readable storage devices as described in claim 39 further including the step of developing the application in an interpreted language.

49. One or more processor readable storage devices as described in claim 41 wherein the step of serializing comprises:

assigning a function ID to each function in the application.

50. One or more processor readable storage devices as described in claim 49 wherein the method further includes:

creating a function ID table associating each function ID with function code.

51. One or more processor readable storage devices as described in claim 49 wherein the method further includes assigning function identifiers to functions within closures.

52. A method of developing an application, comprising:

compiling first object code for an application;

loading the application into a first runtime environment;

creating a serialized representation of a memory space in said first runtime environment;

building second object code using said serialized representation; and

deploying said second object code

53. The method of claim 52 wherein the step of compiling is performed on an interpreted language application.

54. The method of claim 52 wherein the step of creating is performed by calling at least one function from said first runtime environment.

55. The method of claim 52 wherein the step of creating is performed in the same runtime environment as said application.

56. The method of claim 52 wherein the step of creating is performed in a different runtime environment from said application.

57. The method of claim 52 wherein the step of loading the application further includes executing portions of the application marked for execution prior to said creating step.

58. A method, comprising:

receiving from a runtime environment a serialized representation of objects in a memory space of the runtime environment; and

building an object code file using the serialized representation and a compiled object code file used to create the memory space.

59. The method of claim 58 further including the step of: providing the compiled object code file used to create the memory space to the runtime environment prior to said step of receiving.

60. The method of claim 58 further including the step of initializing a serialization process in a separate memory space to create said serialized representation.

61. The method of claim 58 further including the step of initializing a serialization process in the runtime environment to create said serialized representation.

62. A method for delivering an application via a network, comprising:

creating a serialized representation of application objects in a runtime environment;

building an object code file using the serialized representation; and

providing the object code file to a new runtime environment via the network.

63. The method of claim 62 wherein the step of creating includes enumerating a description of each object of the application using reflection

64. The method of claim 63 wherein the new runtime environment is a virtual machine.

65. The method of claim 64 wherein the virtual machine is a Flash renderer.